AMYGDALA $z \mapsto z^2+c$

A Newsletter of Fractals & $\mathcal{M}$ (the Mandelbrot Set)

Issue #26

June 17, 1992

In This Issue...

Busted By The Math Police!, continues John Dewey Jones' explorations in the Fractal Fiction genre, or Math Fiction, as one reviewer called it. John's previous efforts include *The Amygdalan Sects,* reprinted in Amy #25, *Dr. Carl Spring Replies,* in #4, and *The Monastic Sects,* in #15.

Synchronous Orbits by Steve Stoft introduces a novel algorithm with the ability to calculate some fractal images much faster than with current methods. Although I haven't played with it myself, I am greatly impressed with it as a mathematical idea, and look forward to exploration and evaluation of the algorithm by various experimenters.

Stoft's article is followed by reviews of his Fractal WitchCraft program by Dan Lufkin and Leonard Herzmark.

Busted By The Math Police!

— John Dewey Jones

The ugly incident referred to above[1] occurred on the third day of the workshop, at 6:15 AM. I was the only person awake when there came a vigorous knocking on the doors of the Glacier Institute. Two uniformed men stood there, tall, gaunt, their eyes hidden by mirror shades. One of them flipped open his wallet, displaying a silver cardioid.

"Mathematics Police. Mind if we take a look around?"

Without waiting for an answer, the two men brushed past me and went into the seminar room, where they began leafing through the ring binders containing our slides.

1. In Jones's review of the 1991 Fractal Workshop

"What's the problem?", I asked, a little taken aback by all this.

"Obscene materials!", the taller man snapped. "We're looking for obscene materials, and you've sure enough got some vile stuff here!"

"This is a workshop on fractals," I protested, "These slides are all computer-generated."

"That's right! Computer-generated mathporn! Look at this!" He waved a page of Bob's dithered images; several slides dropped to the floor.

"What's wrong with these slides? I don't see anything salacious or suggestive about them."

Both officers paused in their search and looked at me. The shorter man took out his wallet and showed me his silver cardioid.

"You know what this is?"

"Of course: it's the Mandelbrot set."

"That's right. And what does it mean to you?"

"Well, you could view it as an index, a directory to the quadratic Julia sets..."

"Right. Right." The two men grinned wolfishly. "And what does *this* mean?"

CONTENTS

One of them pulled a Cibachrome print from the folder — a beautiful curved shape, suggesting a snail shell or a Roman helmet. The shape stood out against an earth-toned background; its lower part was shaded, giving it solidity and weight.

"It's a hybrid image — you do a few iterations of one formula, then you switch to another one — then the final image has been dithered, see, that's what gives you the texture here..."

"That's not what we asked! What does it *mean?* What mathematical truth does this image illustrate?"

"Oh... none, that I can think of offhand."

"So its only purpose is to titillate, right? To appeal to the senses, rather than the intellect?"

"There's a legitimate place for a sense of beauty in mathematics, I think.... Mightn't we say that an object strikes us as beautiful only because we perceive a hidden order underlying its appearance, even though we may not yet be able to express that order in a theorem?"

They sneered. The taller officer reached under his jacket and brought out a thin silver box, about the size of a paperback book. He put this on the table and touched its top surface. The air above the box shimmered, then a grey image formed. With a slight shock, I recognised it as Bertrand Russell. The two officers stood with heads bowed as the image spoke:

"Mathematics, rightly viewed, possesses not only truth but supreme beauty — a beauty cold and austere, like that of sculpture, without appeal to any part of our weaker nature, without the gorgeous trappings of painting or music, yet sublimely pure, and capable of a stern perfection, such as only the greatest art can show."

As the image faded, the officer replaced the box beneath his jacket. "You see? The beauty he speaks of is the beauty of underlying form, which is perceived by the intellect, not by the senses. This (as he spoke, he slowly tore the Cibachrome print into strips) has no underlying form. It is a travesty, a corpse assembled from unrelated parts, its face painted to give the appearance of life."

"You don't know what you're doing." he continued. "You're not mathematicians. By presenting these daubings in the same forum as images of the Mandelbrot Set itself (both bowed their heads slightly), you cheapen mathematics and confuse the uninitiated. Bit-mapped graphics could be a powerful tool to educate and instruct, but you're making it a carnival sideshow — anything will do, as long as it looks exciting!"

"So what are you planning to do?"

"These images will be destroyed. From now on, anything published in *Amygdala* must be cleared with the Math Police first. For the first year, I think we will restrict you to greyscale images; that should encourage a more responsible approach."

"And suppose I remind you that, under the First Amendment, we can publish what we like?"

"The Mathematics Police are not subject to federal, state or municipal law. We are subject only to first-order logic.

The Slides (S26)

The four slides of this set are all by Robert Hill. It was difficult to select four out of the thirteen remarkable slides I got from Bob at our fractal workshop last summer.

Bob is writing an article about the methods he uses for generating these images, which I expect to publish in Amygdala #27 or 28. In the meantime, I can relay a little of what he told me last summer.

The images are in the "iterative" genre; that is, they involve iterating some functional form for each point in the image plane until some terminating criterion is satisfied.

This involves two different functions of x and y, i.e. new $x = f(x, y)$ and new $y = g(x, y)$ where f and g do not have to be in any particular "analytic" relationship. Bob typically uses the first few terms of the power series for some complex function like $\sin(z)$, and then varies the coefficients to produce interesting effects. He may also iterate on one function for a while, and then switch to another.

I find particularly striking the painterly textures, which are gotten by choosing to color a point according to a random selection based on the "normal" coloring for the point.

For reference, here are my names for the four slides:

A2421: Spider's web.

A2412: Fierce moth.

A2413: Prickly pear.

A2417: Oils on rough canvas.

Synchronous Orbits: The Fastest Mandelbrot Algorithm

— *Steven Stoft*

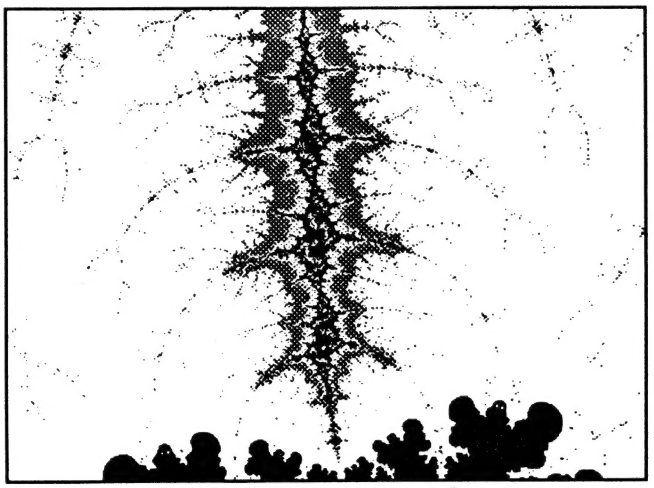
From the beginning, enormous computation times have obstructed the path of research into the Mandelbrot set ($\mathcal{M}$ set). Julia and Fatou might have discovered it in 1918 had they had a desktop computer. Mandelbrot only saw shadows of the set on his Harvard computer, and was forced to travel back to his base at IBM for a closer look. Last May in this newsletter we were warned about an image from the $\mathcal{M}$ set: "Please don't be put off by the ultra-low resolution! Considering that #819 has magnification 9.05×10^{298}, it's a miracle that the authors were able to compute it at all!".

I will now describe what I believe to be the biggest single advance in reducing computation time since Peter Moldave first computed the $\mathcal{M}$ set for Benoit Mandelbrot in 1980. This advance will make it possible for both microcomputers and supercomputers to explore regions that were previously out of reach.

One of the most powerful speed-up algorithms to date relies on following the edges of the dwell bands, thereby avoiding the computation of the pixels interior to those bands. Unfortunately, many, if not almost all, of the more interesting high-magnification images contain large regions where dwell bands are only two or fewer pixels wide. For such narrow bands, edge following provides no speed increase at all.

The new "synchronous-orbit" algorithm described here speeds up not only wide-band fractals but narrow-band fractals as well; in fact it can decrease computation times by a factor of 100 or more. This advantage will increase dramatically as deeper fractals are explored. The narrow-band fractal, "Lightning Strikes at Midnight" (Figure 1), is a case in point. *Fractal WitchCraft*, a program utilizing the synchronous-orbit algorithm, computes this image 338 times faster than *FractInt* and 142 times faster than *Fractal Magic*. The comparison was done on a 386-based PC with a math chip. All three programs used the math chip.

To be more specific, *FractInt* took 25 hours and 50 minutes, *Fractal Magic* took 10 hours 50 minutes, and *WitchCraft* took 4 minutes and 35 seconds!



Lightning Strikes at Midnight
(The ghosts's eyes are bogus)

Program	Time (minutes)	Relative Time
WitchCraft	4.58	1
Fractal Magic	650.	142
FractInt	1,550.	338

Table 1:

But even this algorithm has its limits, and provides much more modest speed gains on what might be called "hyper-narrow-band" images.

In the "Lightning" image, all pixels have dwells above 5,388, but very few have dwells above 5,800. Because the dwell levels have a range of about 400, about half the picture is covered by dwell bands that are two pixels wide or less. In this region edge-following is of no use, but as we will see, the synchronous-orbit algorithm works extremely well.

Parameter	Value
Center, x	–0.1049,7756,7388,0424
Center, y	–0.9270,4136,3780,9322
Suidar (1/r)	1.978×10^{11}
Aspect ratio	4:3
Maximum dwell	10,000
Resolution	640 x 480
Escape radius	2.0

Figure 2: Parameters for "Lightning" Image

No pixel in this image leaves the radius-2 circle ($|z| < 2$) during the first 5,388 iterations. This sug-

gests that the orbits of these points may behave quite non-chaotically relative to each other for many iterations. Also remember that this image is quite small, about 10^{-11} on a side, so we would expect the orbits of all points within the image to start out in parallel. After the first iteration we can be sure that the whole rectangular set of about 300,000 points has moved together to a new location, where they are again arranged in a rectangle of almost exactly the same shape, but rotated and often expanded. This is the central insight of the synchronous-orbit method.

Let us re-state the concept more carefully. The *orbit* of a point z is the series of locations in the complex plane that it occupies as it is transformed by $z \rightarrow z^2 + c$ repeatedly. Let's consider four points arranged in a very tiny square. When these four points are transformed for the first time, their new values will be almost as close together as the original points were, and they will still form an almost perfect square. As it turns out, their orbits will remain parallel like this for thousands of iterations. That is what's meant by *synchronous orbits*.

In essence, the algorithm works like this: we keep track of the four corners of a square that more than covers the entire image, under the transformation $z \rightarrow z^2 + c$. We follow those corners until they begin to deviate slightly from a perfect square. During this phase we are doing about 60,000 times less work than if we followed each of the 300,000 pixels, including the work of the deviation test. When the deviation becomes large enough to represent an error of say 1/10 pixel in the original image, we divide the square into 16 smaller squares by non-linear interpolation. Because these squares are smaller, they will not have suffered as much distortion, so we can continue our iterations on each square separately. This time each square has only about 20,000 pixels, so we are doing about 4,000 times less work than we would by tracking every pixel.

As the squares continue to become distorted we continue to divide them into smaller squares. Finally we will come to a square so small (4×4 pixels) that it is easier to track individual pixels than the whole square. So that is what we do, resorting to the standard Mandelbrot algorithm to compute the orbits of individual pixels. For a wide-band image we may never have to resort to individual pixel calculation, because a whole square of pixels may leave the radius-2 circle together, undistorted, but for narrow-band images we always finish up with the same tedious pixel-by-pixel calculations. For a fractal like "Lightning" this will entail only a few iterations on any given individual pixel, in spite of its dwell being above 5000.

You now know the essential secret of the synchronous-orbit algorithm for computing fractals, but this algorithm harbors secrets perhaps not to be discovered for years to come. Here is a hint about one of its undiscovered secrets. To make this algorithm work successfully one must detect just when a square of pixels is about to become too distorted, making it necessary to break it into smaller squares. I discovered a part of a rule that seemed like it just might work, but to make the rule complete I need to find the right values for several of its parameters. I modified these parameters for a few dozen images, but was never able to prove that either the rule or the parameters were correct. Luckily, the rule has worked without failure on thousands of fractal images, but I don't know why.

A more sophisticated version of the synchronous-orbit algorithm (to be discussed in a future issue of Amygdala) has another wonderful, mysterious, and accidentally discovered property. Late one night, after making a particularly significant improvement, I wanted to test the algorithm's ability to reach high magnifications. So I went out to a random point on the $\mathcal{M}$ set's "beetle snout" and began zooming in. Since I was using double-precision floating-point arithmetic with about 15 decimal places of accuracy, I expected that the image would start to disintegrate some time before a magnification of 10^{14}, since beyond this magnification, two adjacent pixels on the x axis are likely to be given the same x values simply because of rounding errors. Again and again I zoomed in, each time enlarging a small speck of the image by more than 25×. Soon I realized that I must be well beyond the double-precision barrier, so I wondered if I was actually zooming in, or if some bug was just shifting me around in a deceptive way. To find out, I made a sequence of partial zooms, so I could watch the landscape change more incrementally. Sure enough, I saw the image enlarging just as it should! Nothing funny going on except all this

impossible zooming. When I finally stopped and checked a window to see my depth, it read: Magnification = 3.511×10^{84}!! This was some kind of witchcraft!

I have since checked this phenomenon by using 7-place arithmetic, going to a magnification of 10^{14}, and then recomputing the same image with double-precision arithmetic. The image is the same.

As I write this, I am again amazed by this discovery, and stop to check that it still really works. The program, now so much friendlier than on that first excursion, reads out my magnification as I go, and adjusts the maximum computed dwell level automatically. Each zoom of about 25× takes less than five seconds, and as I pass 10^{50}, I slow down again to check if it's really working. I notice a quirk caused by the lack of precision: I can no longer control the x-value of the picture's center; but the zooming is working perfectly. At 2.2×10^{100} I stop and ask it to draw a full screen picture, which takes just 1 minute and 6 seconds. With a program computing every pixel with 100-digit accuracy, this would have taken nearly a day. Then I remember, there is no need to zoom: I can set the magnification directly. Calling up the appropriate window, I set magnification = 10^{305}; that's about it for double-precision, and it's just past the record reported here last May. I tell it to draw: it hesitates. I know it's iterating a square that covers the whole image, then breaking off a corner, and a corner of a corner, until it gets down to the pixel level. The suspense is quickly over and it starts to draw. Once it starts, it doesn't hesitate, and it finishes the screen in about a minute. This is perhaps the smallest image ever seen.

This technique has one serious limitation: you can only compute images around a point which can be specified by numbers of the precision used for tracking. When you get to a high enough magnification such points seem very far apart. They almost certainly lie either too far inside or too far outside the $\mathcal{M}$ set to be of any interest — unless of course you are exactly on the beetle's snout, as I happened to be that first night when I zoomed past the natural limits of computation.

Fractal Witchcraft And Fractint — Gatorade vs. Bouillabaisse

— Dan Lufkin

Perhaps it's a little late in the day to be reviewing FractInt for the readers of Amygdala. For a couple of years now FractInt has been the definitive DOS fractal program and there are probably only a few recalcitrant Macintoshians and Amigans who haven't had a chance to run it. Nevertheless, as the newspaper of record in the Mandelbrot universe, Amygdala needs to chronicle the Age of FractInt and deal with its impact on other fractal software.

Since 1988 FractInt has been taking shape under the guardianship of the Stone Soup Group, a loose conglomeration of genius programmers and fractoids who collaborated to create one of the most remarkable pieces of software ever written. Not only is FractInt unique as a fractal program, it is unusual in the way in which it was created.

Stone Soup

For those of you who came in late, the Stone Soup Group takes its name and its inspiration from the folk tale about the peddler who comes into a village in the midst of famine. The peddler sets up a big iron pot in the square, pours in some water, and then carefully unwraps a large black stone from his pack. When the water is boiling, he drops in the stone, then sits back with his bowl, ostentatiously preparing to enjoy a hearty meal. One of the villagers timidly asks if he really can make soup from a stone. "Of course," replies the peddler, "though it's really better with a little cabbage in it." The villager has a head of cabbage hidden away, so he brings it out, agreeing to add it to the pot in exchange for a share of the soup. Soon other villagers have contributed a little bacon, some turnips, a few onions, a piece of salt beef, potatoes, and everything else needed to make a big pot of really good soup. They hungrily share it out and realize that there was actually enough food in the village for everyone once the peddler had contributed the essential ingredient.

In the FractInt soup the catalyst was the program FRA386 written by Bert Tyler in 1988 to take advantage of the new capabilities of the 80386 for fixed-point calculation. Tyler's approach was so successful that cognoscenti always refer to FRA386

as "blindingly fast." Mathematician Tim Wegner got hold of the public-domain program, added support for improved graphics, 3-D display options, and collaborated in laying out FractInt's basic architecture. Collaboration took place via CompuServe's COMART forum, where Mark Peterson soon chimed in with the periodicity-checking algorithm, a formula parser and improved assembly-language subroutines. Pieter Branderhorst added an improved user interface and speeded up the code still more. Perhaps a hundred other villagers from the COMART electronic village have contributed ideas, bits of code, snazzy color schemes, sound effects, fractal types, graphics drivers and other bells and whistles.

How's the Soup?

For most programmers the amazing thing about FractInt is the way its seamless structure encloses a grab-bag of routines, some masterpieces of precision assembly coding and some brute-force C-language crowbars. It's easy to be a kibitzer on FractInt because the source code is downloadable from COMART (section 15) as is the executable program, the documentation (not as extensive as it once was, but not really needed any more), dozens of color palettes, GIF images, related programs (including a Windows version, WINFRAC). All of this is freeware. The motto of the Stone Soup Group is "Don't want money. Got money. Want admiration."

The best way to understand FractInt is to buy the official FractInt scripture, The Waite Group's Fractal Creations by Timothy Wegner and Mark Peterson (Waite Group Press, 100 Shoreline Highway, Suite A-285, Mill Valley, CA 94941, $34.95, 1991, ISBN 1-878739-05-0), which comes with a 5 1/4" disk with everything in the pot on it. FC has not only a complete guide to the riches of FractInt, but also an excellent discussion (the best in print, in my view) of the mathematics and programming of fractals.

One of the great strengths of FractInt is its use of text files to configure the program and set parameters. I used to think that control files ought to have been abandoned with the punch card, but now I appreciate the power that a robust text parser can bring to an application. When FractInt starts up, it looks for a file SSTOOLS.INI in the path and reads its start-up variables and commands from it. This is like the WIN.INI file in Windows and, like WIN.INI, it can be a powerful tool if you learn how to use it.

A FractInt run can be saved as a .PAR file, given any fanciful title you choose, along with a line of explanation, and sent via modem to your fellow fractoids at a couple hundred bytes each, including a 256-color palette. The palettes themselves are stored as .MAP files and can be swapped and modified very readily. FractInt itself can be run with dozens of command-line options from a .BAT file to control all sorts of program variables. (It will even write its own .BAT file to record the current parameters for a run.) You can write a .KEY file to run a slide show. If you create a separate FractInt .CFG file, you can set FractInt up to handle nearly any graphics adapter ever made. (It took about 10 minutes from scratch to get my ATI Wonder+ working with 1 MB memory.) FractInt will even print your fractal on a variety of laser, ink-jet and dot-matrix and PostScript printers. (Including 13 different halftone patterns for PS!)

Good Meat at the Bottom

An even deeper capability of FractInt is its formula compiler which interprets (well, OK; it isn't technically a compiler, I guess) algebraic expressions into program instructions. You can set up a formula file of your own with the extension .FRM or you can add fractal types to the 60-odd already in FractInt.FRM. The parser reads the initial condition, the iteration formula, and the escape condition from the formula file and applies all the other capabilities of the program to displaying the result. A classical Mandelbrot fractal, for example, would simply be

```
Mandelbrot { c = z = pixel : z = (z*z)+c, |z| <= 4 }.
```

With this capability it is relatively straightforward to produce displays of the Lorenz equations and the like. An extension to the formula compiler operates on Lindenmayer L-systems that model the growth of plants with a drawing "turtle." Koch curves and Sierpinski gaskets are two of the fractals that can be constructed from L-systems. The compiler can also handle the parameters for Barnsley Iterated Function System images (the famous fern pattern is one) and Ackerman exponential function fractals.

FractInt stores its images in CompuServe Graphics Interchange Format (GIF) and can also display general GIF files. I use FractInt to show satellite remote sensing images.

Can Man Live On Soup Alone?

Not to put too fine a point on it, FractInt is more a phenomenon than a program. It's hard to see how any other fractal software could offer much competition, especially since there are many more features of FractInt that I don't have space to discuss here. One drawback to the Swiss Army Knife approach to software, though, is that you can easily lose sight of your original goal and become lost in the embellishments.

The fact of the matter is that FractInt is fast, but no longer blindingly fast. It incorporates such speed-up algorithms as the Mariani technique (see Amygdala #4), edge tracing, and orbit periodicity checking. There are regions in fractals, though, in which steep gradients in high-dwell areas mean that none of the "classical" techniques (Mariani is five years old) gives much improvement in speed.

Hey, Mac, What Are Those Witches Cooking?

Along comes Steven Stoft with Fractal WitchCraft, a program that has to be called "stupefyingly fast." Stoft's algorithm has been discussed in Algorithm and here in Amygdala as well. (For those of you who are reading a copy of just this page, it involves tracking the orbits of the four corners of an image square and subdividing the square only when it becomes distorted.) In some parts of the complex plane, this speeds up the creation of the image by a factor of several hundred.

When I first started out to write this review, I thought it would be a simple matter to stage a race between Fractal WitchCraft and FractInt and report the respective times for some of the staple images in The Beauty of Fractals. In practice, though, I found that differences in color palettes, dwell breakpoints and other factors made it very difficult to produce images that were truly comparable. Given enough time to fine-tune each program's image parameters, I'm sure that identical screens would result. Spending a month fiddling in order to save a few hours of computer time, however, is precisely the mind-set that gives computer hacking its bad name.

Waiter, Why Can't this Soup Fly?

At the risk of getting whiplash from the sudden change of metaphor, it's better to think of Fractal WitchCraft as a specialized tool, a sort of helicopter for fast reconnaissance on the Mandelbrot set, with FractInt as the all-terrain truck that you can bring in later for the heavy-duty earth moving work once you've discovered a likely area to explore. FractInt's ultra-powerful graphics controls and ability to work with non-Mandelbrot/Julia fractals more than make up for its lack of speed if you want to produce an eye-popping image that you can photograph or upload to your friends in .PAR or .GIF form. On the other hand, if you're searching around for something in there that looks like, say, a Studebaker logo, you can cover a lot complex territory with Fractal WitchCraft. Each program has its strengths and with both of them in your files you can do things that neither one alone could handle well.

Sources

Fractal WitchCraft is essentially shareware with a rather complex method of discounts and incentives for distribution. Otherwise it costs $29.95 from SuperNatural Graphics, P.O. Box 9185, Berkeley, CA 94709. FractInt is freeware, to be found on many bulletin boards, CompuServe's COMART forum, or from me (Dan Lufkin, 303 West College Terrace, Frederick, MD 21701) for $2.00 for a 3.5 or 5.25-inch high-density disk.

A Review of Fractal WitchCraft

— A letter from Leonard Herzmark

Dear Rollo,

I'm responding to your pressing request for a review of Fractal WitchCraft. I loaded it on my machine and while I was able to make it work, the documentation is so poor, I am unable to understand how to save a file with its parameters. I was able to zoom in on one of Steven Stoft's stock pics and generate and save one blowup, but certainly the speed wasn't anything to speak of. It took over

an hour and a half to fill the screen with a VGA (640×480) image in 12 colors. The original from which I did the blowup looked like a Julia, but couldn't tell as couldn't read the parameters or type. I tried everything in the meager documentation and couldn't get it to display those data. In case you speak with Steve Stoft, you might tell him of this report and advise him that the name of the original file was SPARKLE.PCX, and the one I generated from it is RATTLER.PCX. I'm enclosing a printout of the directory showing the size and minimal other information on both. You might ask him what type of fractal that file was generated from and what its parameters are. If he would like to direct that info to me I'll see what the time would be on other programs to duplicate it, provided of course that I have some which cover that type. I have Julia and Mand..., in different programs, and of course FractInt, so can possibly make comparisons.

I grant that Steve can generate a Mandelbrot rather fast in the miniature, but to do a full Mandelbrot VGA is much slower in his program than in FractInt. Of course, the problem might be that I can't read his documentation and understand it. I couldn't find his timer mechanism, so I had no sure way of reading the time to run.

Sincerely, Len

MathVISION

— George Zopf, reviewer

MathVISION
Seven Seas Software
PO Box 1451 Port Townsend, WA 98368
Three disks and manual, $197.00

Since its inception in 1985, the Commodore Amiga has been well served with a considerable variety of mathematical software for color display of fractals and the Mandelbrot and Julia sets. Although the bulk of these programs have appeared in the public domain, an especially lively field among Amiga enthusiasts, there is a good handful of commercial programs. One of the earliest to appear was Doug's Math Aquarium, from Seven Seas Software. Largely the work of Doug Houk, the program displayed functions of one and two variables as color contours. MathVISION is a considerably reworked and enhanced avatar, well deserving its subtitle of "A Tool for Mathematical and Scientific Visualization".

MathVISION's price of $197 would not startle Macintosh users, but may seem steep to Amiga owners: nevertheless, it gives good value. The three-disk package includes two versions of the program: one for single precision (Motorola FFP math package) and one for double precision (IEEEdbl). I tested both versions on an Amiga 2000 (Motorola 68000 CPU at 7.14 MHz) and an Amiga 3000 (Motorola 68030 CPU with Motorola 68882 FPU, running at 25Mhz). Anyone who has gone through a sequence of zooms on the Mandelbrot set knows that computation-intensive operations take a long time unless an accelerated chip (68020 or 68030) is installed; Amiga owners have the additional advantage of setting the process going, and then, through multitasking, of turning to other computer activities, returning from time to time to see how the plot is progressing. Use of lower-precision math speeds up the rendition, of course, but limits depth of iteration and zoom. I found that MathVISION on the Amiga 3000, using double-precision math, gave satisfyingly brisk visualizations, almost worth watching as they develop; on the Amiga 2000, even with single-precision math, a certain amount of boredom sets in.

A remarkable number of standard functions are available: basic operators, logarithmic, trigonometric, hyperbolic trigonometric, boolean, rounding and clipping, as well as a full set of environmental functions associated with the display. MathVISION comes with a full complement of accessory goodies, notably its "Hooks" and ARexx interface. The "hooks" are auxiliary programs "attachable to the main MathVISION program to supply special functionalities. There is a Mandelbrot "hook" to optimize computation of Mandelbrot and Julia sets, a "Divide and Conquer" hook used to speed up the display of contour plots, a sound hook allowing four-channel sound to be associated with plotting, and a "Focus" hook, especially designed for zooming into the Mandelbrot set, to mention only a few.

Although I have not explored it with MathVISION, the provision of an ARexx interface may well be its most exciting possibility. ARexx, for non-Amiga users, is the Amiga port of the inter-

preted language Rexx, first developed for IBM mainframes. ARexx, though a "full-feature" language, is especially suited as a scripting-and-macro language, capable of communicating with and controlling any program provided with ARexx ports (a simple matter). An ARexx program can put another program through its paces. For example, a program could be written to lead a Mandelbrot set display program through an arbitrary series of zooms: producing one's own version of "Nothing But Zooms" automatically. (ARexx is part of the Amiga 3000 package, and is available commercially for earlier models.)

MathVISION has its idiosyncracies (Someone once said, "One mathematician's notation is another mathematician's bitter experience"), but should serve well as gun and camera for both the casual and devoted mathematical explorer.

Books and Software

— from the Springer Newsletter, Computer Science #1/92

The Beauty of Fractals Lab, Graphics Software for the Macintosh, Version 1.0; H.-O. Peitgen, H. Jürgens, and D. Saupe; written by T. Eberhardt and M. Parmet.

Interact with and explore the Mandelbrot and Julia sets with this diskette. Find your own zooms and artistic color maps. You can easily switch between 2D, 2.5D and 3D renderings. The algorithms are very fast in the sense that the predominant shapes of your images emerge quickly. Immediately you can choose to initiate your next blow-up, or you may wish to save window coordinates for later processing of a fully computed high resolution image. These images can be inserted into your favorite drawing program or word processor allowing you to create beautiful prints.

Hardware requirements: Macintosh II computer, running system 6.0.2 or later, with color or grey scale monitor (256 colors/shades) and 1MB memory (2MB or more preferred).

$49.00 (1991).

Fractals and Disordered Systems, A. Bunde and S. Havlin.

This well-illustrated volume presents modern concepts and tools for current research on fractals and disordered systems, discussing in great detail the effects of disorder on mesoscopic scales (fractures, aggregates, colloids, surfaces and interfaces, glasses, and polymers) and presents tools to describe them in mathematical language.

The ten contributions by experts such as B.B. Mandelbrot, H.E. Stanley, and J.K.Kjems comprise a pedagogical and systematic survey of experimental and theoretical approaches, e.g., scaling theory, computer simulations, and practical laboratory experiments. This book is useful both as a graduate level text and a professional reference.

350 pp., 163 illus. (20 color plates), Hardcover $59.00 (1991).

Authors

Professor *John Dewey Jones* was born in England, and sometime later acquired a degree in mathematical physics. After an interlude teaching high school in Kenya, he took a doctorate in engineering and went to work for General Motors Research Laboratories, where he became interested in fractals.

He now works and teaches in the School of Engineering Science at Simon Fraser University, Canada, where he has just achieved tenure.

His current research interests include Stirling engines, applications of artificial intelligence to engineering, and fractals.

Steven Stoft received his B.S. in math in 1969, and his Ph.D. in economics in 1982, both from U.C. Berkeley. He has taught at Boston University and U.C. Santa Cruz, but currently does energy policy research for Lawrence Berkeley Laboratory.

Circulation

As of 17 June 1992 Amygdala has 571 subscribers, 209 of whom have the supplemental color slide subscription.

Renewal

For 46 of you subscribers out there this is the last issue of your current subscription. I urge you to use the enclosed form to renew your subscription promptly to avoid missing anything.

MERZ

Fractal software, publications, and other stuff available from individuals and small companies.

Dada was a movement of artists and writers in the early 20th century who exploited accidental and incongruous effects in their work and who programmatically challenged established canons of art, thought and morality.

Dada was originally called *Merz* — the innermost syllable of *Commerzial.*

Merz is an issue-by-issue directory of products and services provided to our readers by individuals and small companies or organizations. We will accept ads for software of educational or recreational value, pamphlets, books, videotapes, clubs, organizations, T-shirts, fractal images, calendars, etc.

To place a Bulletin Board item, send camera-ready copy to fit into one or more rectangles — or fill out your copy by hand and we will typeset it. Be sure to include all relevant information, such as address and conditions of distribution. We will accept 1x1 areas (like top right), 1x2 or 1x3 areas (like this) 2x2 or 2x5 (full page).

Ads cost 30.00 per issue per basic rectangle. There is a 10% discount for 2 or more insertions placed at the same time (not necessarily for the same issue). Send to:

Amygdala Ads • Box 219 • San Cristobal, NM 87564
Telephone: 505/586-0197

SPECIAL ON BACK ISSUES

During July and August you can get two or more back issues of Amygdala for 20% off, plus free postage in US!

US: $2.00 per issue, including postage

Elsewhere: $2.00 per issue, see Order form for postage

Amygdala • Box 219 • San Cristobal, NM 87564• USA • (505) 586-0197

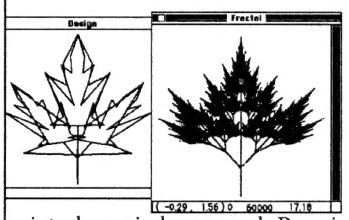
Fractal WitchCraft

Great "Slide" Shows. Easy Flexible Color Control. Context sensitive help. PCX files. Adapts to 286/386, 80x87 automatically. You need: PC with VGA or EGA. $14.95;

SuperNatural Graphics / PO Box 9185

Berkeley, CA 94709

Announcing ***Fractal Attraction***™ for the Macintosh®



Design your own fractals using iterated function systems (math degree not required!). Scale, rotate, shear, translate transformations all with the mouse. Interactively apply the collage theorem. Features color rendering. Fractal can be printed, copied or saved. Requires MacPlus or up. Write or call for more information. To order send $49.95 to Sandpiper Software, P.O. Box 8012, St. Paul, MN 55108. (612) 644-7395

FRACTAL COSMOS PORTFOLIO

Featuring 14 fractal color images by Rollo Silver, Ken Philip, John Dewey Jones, Ian Entwistle, etc. (The 1991 Fractal Cosmos Calendar)

$4.95 plus Postage & handling (see Amygdala price list)

Amygdala • Box 219 • San Cristobal, NM 87564• USA • (505) 586-0197

ART MATRIX

"Nothing But Zooms" video, Prints, FORTRAN program listings, postcard sets, slides.

Send for FREE information pack with sample postcard.

Custom programming and photography by request.

Make a bid.

PO Box 880 • Ithaca, NY 14851 • USA

(607) 277-0959

Mention Amygdala, Please!

If you place an order for a product as a result of seeing it advertised here, please let the advertiser know!

AMYGDALA, Box 219, San Cristobal, NM 87564
505/586-0197
See ORDER FORM for subscription information